PCT WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 7 : G06F 17/30 — A1

(11) International Publication Number: **WO 00/57303**

(43) International Publication Date: 28 September 2000 (28.09.00)

(21) International Application Number: PCT/US00/06331

(22) International Filing Date: 9 March 2000 (09.03.00)

(30) Priority Data:
09/273,606 22 March 1999 (22.03.99) US

(71) Applicant: MATTEL, INC. [US/US]; 333 Continental Blvd., El Segundo, CA 90245 (US).

(72) Inventor: YAMAMOTO, Keith, K.; 38741 Huntington Circle, Fremont, CA 94536 (US).

(74) Agent: KRESS, Hugh, R.; Howrey Simon Arnold & White, LLP, 750 Bering Drive, Houston, TX 77057-2198 (US).

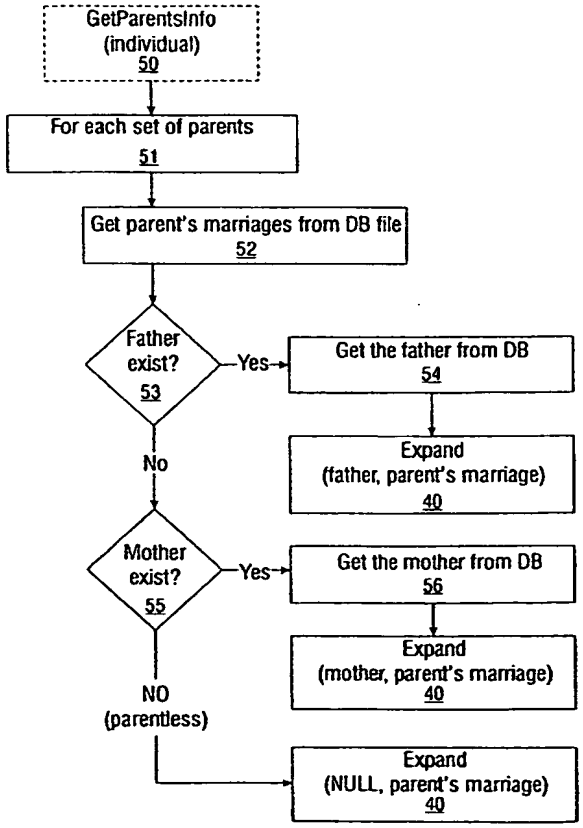
(81) Designated States: AE, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, CA, CH, CN, CR, CU, CZ, DE, DK, DM, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, UZ, VN, YU, ZA, ZW, ARIPO patent (GH, GM, KE, LS, MW, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

Published
With international search report.
Before the expiration of the time limit for amending the claims and to be republished in the event of the receipt of amendments.

(54) Title: METHOD AND APPARATUS FOR GENERATING AN ALL-IN-ONE FAMILY TREE

(57) Abstract

A method for organizing a conventional family tree database file (40 and 50–56) to allow a user to simultaneously display all ancestors and descendants of the file. A family tree (40 and 50–56) is established by traversing the file multiple times to associate all parents (50), spouses, and children of each individual with the individual's respective parents (50), spouses, and children named in the file.



FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AL	Albania	ES	Spain	LS	Lesotho	SI	Slovenia
AM	Armenia	FI	Finland	LT	Lithuania	SK	Slovakia
AT	Austria	FR	France	LU	Luxembourg	SN	Senegal
AU	Australia	GA	Gabon	LV	Latvia	SZ	Swaziland
AZ	Azerbaijan	GB	United Kingdom	MC	Monaco	TD	Chad
BA	Bosnia and Herzegovina	GE	Georgia	MD	Republic of Moldova	TG	Togo
BB	Barbados	GH	Ghana	MG	Madagascar	TJ	Tajikistan
BE	Belgium	GN	Guinea	MK	The former Yugoslav Republic of Macedonia	TM	Turkmenistan
BF	Burkina Faso	GR	Greece			TR	Turkey
BG	Bulgaria	HU	Hungary	ML	Mali	TT	Trinidad and Tobago
BJ	Benin	IE	Ireland	MN	Mongolia	UA	Ukraine
BR	Brazil	IL	Israel	MR	Mauritania	UG	Uganda
BY	Belarus	IS	Iceland	MW	Malawi	US	United States of America
CA	Canada	IT	Italy	MX	Mexico	UZ	Uzbekistan
CF	Central African Republic	JP	Japan	NE	Niger	VN	Viet Nam
CG	Congo	KE	Kenya	NL	Netherlands	YU	Yugoslavia
CH	Switzerland	KG	Kyrgyzstan	NO	Norway	ZW	Zimbabwe
CI	Côte d'Ivoire	KP	Democratic People's Republic of Korea	NZ	New Zealand		
CM	Cameroon			PL	Poland		
CN	China	KR	Republic of Korea	PT	Portugal		
CU	Cuba	KZ	Kazakstan	RO	Romania		
CZ	Czech Republic	LC	Saint Lucia	RU	Russian Federation		
DE	Germany	LI	Liechtenstein	SD	Sudan		
DK	Denmark	LK	Sri Lanka	SE	Sweden		
EE	Estonia	LR	Liberia	SG	Singapore		

BNSDOCID: <WO_____0057303A1_I_>

METHOD AND APPARATUS FOR GENERATING AN ALL-IN-ONE FAMILY TREE

FIELD OF THE INVENTION

This invention relates generally to an automated method for organizing a database file, and more particularly, to an automated method for organizing and displaying a family history database file so as to identify not only ancestors and descendants of an individual, but also in-laws, step-children, multiple spouses, and un-related individuals.

BACKGROUND OF THE INVENTION

Genealogy involves the study of a family lineage or history. Typically, a family history is obtained by extensive research of an individual's ancestors and descendants. Provided that there is an established marital or blood association between the related members of the family history, the information obtained can be compiled to form a family tree chart for visual inspection.

Various family tree charts have been established over the years to aid in viewing the descendants or ancestors of a family history. In recent years, various software manufacturers have developed programs that have transposed the same written charts for a computer display. More specifically, these genealogy programs provide the user with the ability to graphically display the direct ancestors or descendants of an individual whose name is stored in a database file along with the names of his or her ancestors and/or descendants. In other words, by means of lines or arrows drawn between relatives of successive generations, a computer can organize the family history database file and display the relationships of those relatives in the direct line of descent, namely an individual's children, grandchildren, and so on, and his or her parents, grandparents, great-grandparents, and so on. However, it is believed that no current genealogy program allows a user to simultaneously display both ancestors and descendants of a family history database file. Moreover, it is believed that no presently existing program can currently display the related family history of in-laws, step-children, multiple spouses, and individuals that can not be clearly or directly linked to a particular family history.

Many times, available information for a family tree will not provide a clear link between all members of the family. For example, in-laws, step-children, multiple spouses, and even un-related individuals may be discovered, but more than likely, will

not be easy to associate to the particular family ancestors or descendants. Consequently, it would be advantageous to be able to organize and simultaneously display all ancestors, descendants, and/or individuals of a family history file to provide an all-in-one family tree.

SUMMARY OF THE INVENTION

In accordance with one aspect of the present invention, a method is provided for organizing a database file to develop a family tree chart. In one embodiment, the method comprises the following steps: (a) finding an individual whose name is contained in the database file; (b) creating a tree matrix node for the individual; (c) expanding the individual to identify his or her parents named in the database file; (d) expanding the individual to identify his or her spouse(s) named in the database file; (e) expanding the individual to identify his or her children named in the database file; and (f) repeat above steps (a) through (e) for all individuals named in the database file.

In accordance with another aspect of the instant invention, a system architecture is provided for establishing an association between a plurality of nodes of a family history database file. The system comprises the elements of: a first one of the plurality of nodes having stored information; the stored information being structured to establish the association between at least a second one of the plurality of nodes; wherein the association for the remaining plurality of nodes being established by repeatedly traversing the family history database file to recognize distinctive elements of the stored information of at least one of the plurality of nodes not already associated to at least one of the plurality of nodes.

In another aspect of the instant invention, a program storage device encoding a machine-readable copy of program instructions is provided for the process steps of: (a) create a tree matrix node for an individual; (b) expand the individual to identify his or her parents named in the database file; (c) expand the individual to identify his or her spouses named in the database file; (d) expand the individual to identify his or her children named in the database file; and (e) repeat above steps (a) through (d) for all individuals named in the database file.

BRIEF DESCRIPTION OF THE DRAWINGS

The foregoing and other aspects and advantages of the invention will become apparent with reference to the following detailed description of specific embodiments of the invention, when read in conjunction with the drawings, wherein:

Figure 1 illustrates a block diagram of a computer system suitable for use in implementing the present invention;

Figure 2 illustrates a block diagram showing components of the processor chassis in the computer of Figure 1;

Figure 3 illustrates a general flow chart of one implementation of the inventive process;

Figures 4A - 4F are high level flow charts of the inventive process of Figure 3;

Figures 5A - 5H illustrate an example of how a basic family tree matrix having ancestors and descendant can be developed in accordance with the flow charts of Figures 4A - 4F;

Figures 6A - 6C illustrate an alternative embodiment of the invention in which the basic family tree matrix of Figures 5A - 5H is expanded to include individuals with multiple spouses;

Figures 7A - 7B illustrate still another embodiment of the invention in which the family tree matrix of Figures 6A - 6C is further expanded to include step-children; and

Figure 8 illustrates still another embodiment of the invention in which the family tree matrix of Figures 7A-7B is further expanded to include unrelated individuals.

While the invention is susceptible to various modifications and alternative forms, specific embodiments thereof have been shown by way of example in the drawings and are herein described in detail. It should be understood, however, that the description herein of specific embodiments is not intended to limit the invention to the particular forms disclosed, but on the contrary, the intention is to cover all modifications, equivalents, and alternatives falling within the spirit and scope of the invention as defined by the appended claims.

DETAILED DESCRIPTION OF SPECIFIC EMBODIMENTS OF THE INVENTION

Illustrative embodiments of the invention are described below. In the interest of clarity, not all features of an actual implementation are described in this specification. It will be appreciated that in the development of any actual embodiment, numerous implementation-specific decisions, engineering and programming, must be made to achieve the developers' specific goals, such as compliance with system-related and business-related constraints, that will vary from one implementation to another. Moreover, attention will necessarily be paid to proper engineering and programming practices for the environment in question. It will be appreciated that such a development

effort might be complex and time-consuming, but would nevertheless be a routine undertaking for those of ordinary skill in the field of computer programming having the benefit of this disclosure.

An Apparatus for Practicing the Invention

Figure 1 illustrates a computer 10 suitable for implementing the present invention. The computer 10 may be a desktop computer of the well-known personal computer class, which are sold by many computer vendors and/or manufacturers, such as, for example, Compaq Computer Corporation, Houston, Texas. However, those of ordinary skill in the art will appreciate that the invention is not limited to the personal computer class of machines, and other types of computers, such as palm or laptop computers may also be employed.

Computer 10, in the embodiments illustrated in Figures 1 and 2, includes a processor chassis 12 in which is disposed a central processing unit ("CPU") 13 that includes a motherboard and a variety of other circuit boards, none of which being separately shown. Computer 10 also provides a program storage device, such as the disk (not shown) of a hard disk drive 14. The disk of the hard disk drive 14 may be used to store software implementing the method in its various embodiments as disclosed below. Alternatively, the software may be stored on other program storage devices such as a floppy disk 16 inserted into a floppy disk drive 18 or an optical disk 20 inserted into the optical disk drive 22. A display 28, a mouse 30, and a keyboard 32 assist the user when interacting with the software implementing the invention and a database stored in the personal computer 10, examples of which will be described below.

Figure 2 conceptually illustrates computer 10 and some of its various components, including memory 36 of CPU 13. Memory 36 may include both read-only memory ("ROM") and random access memory ("RAM") coupled to the CPU 13. Also coupled to CPU 13 are display 28, floppy disk drive 20, optical disk drive 22, hard disk drive 14, mouse 30, and keyboard 32. All of these components can be implemented in a conventional manner.

Some portions of the detailed descriptions below are presented in terms of software-implemented methods or algorithms, and/or symbolic representations of operations on data bits within a computer memory. These descriptions and representations are the means by which those skilled in the art most effectively convey

the substance of their work to others skilled in the art. A software implemented method and/or algorithm is here, and is generally, conceived to be a self-consistent sequence of acts leading to a desired result. The acts require at some level physical manipulations of physical quantities. Usually, though not necessarily, these quantities take the form of electrical or magnetic signals capable of being stored, read, scanned, transferred, combined, compared, and otherwise manipulated. It has proven convenient at times, principally for reasons of common usage, to refer to these signals as bits, values, elements, symbols, characters, terms, numbers, or the like.

However, all of these and similar terms are to be associated with the appropriate physical quantities and are merely convenient labels applied to these quantities. Unless specifically stated or as may otherwise be apparent from the descriptions herein, terms such as "processing" or "computing" or "calculating" or "determining" or "displaying" or the like, refer to the action and processes of a computer system, or similar electronic computing device, that manipulates and transforms data represented as physical (electronic) quantities within the computer system's registers and memories into other data similarly represented as physical quantities within the computer system memories or registers or other such information storage, transmission or display devices.

In one embodiment, the invention is implemented as a computer software application, hereinafter referred to as the "All-In-One" application, suitable for execution on "conventional" personal computers ("PCs"). "Conventional" PCs are currently nominally characterized as being based on a 133- to 450-Mhz Pentium™ class of microprocessor platform with, in a typical configuration, 32 or more megabytes of on-board random access memory ("RAM"), a one or more gigabit capacity hard disk drive mass storage unit, and a 28.8 kilobit-per-second (kbps) or faster modem with mass storage unit, and a 28.8 kilobit-per-second (kbps) or faster modem with associated software for providing connectivity to the Internet. Of course, as discussed above, a "conventional" PC-class of computer will also have standard peripheral equipment, including a keyboard, mouse (or other cursor control device), and graphics display monitor.

Graphical User Interface

Because the All-In-One program could become too big to be practical if all family history is included in the tree matrix, the graphical user interface ("GUI") operating system to implement one variant of the embodiment will control most of the layout

settings for the tree. However, the user will preferably have control over many customization option, including, without limitation:

(1) whether or not to display unlinked individuals;

(2) maintaining the birth order of children within a family, or allowing "opportunistic" ordering;

(3) whether or not to display box lines and border styles and text font and style; and

(4) number of generations of ancestors and descendants to show.

The "All-In-One" tree matrix is preferably saved as its own view, with its own styles.

The Method in Accordance with One Embodiment of the Invention

Figures 3 and 4A-4F illustrate basic and high level pseudo code corresponding to a method for generating family tree data in accordance with one embodiment of the invention, respectively. In the presently disclosed embodiments, a fixed or pre-existing database file of a family history is stored on a storage medium selected from the group including: a hard disk drive 14, a floppy disk 16, an optical disk 20, and memory 36 (see Figures 1 and 2).

The present invention relates generally to facilitating the generation of a "family tree" from which the interrelationships between members of a family can be observed. In particular, the present invention relates to a method and apparatus for deriving a so-called family tree matrix from a collection of data about a family's history.

As used herein, the term "family history" is intended to include information about the family lineage of a primary individual, such as all known ancestors and descendants. A "family history" may also include information about in-laws, step-children, multiple spouses, and even un-related individuals that may or may not be related by a common bloodline to the primary individual.

In one embodiment, the present invention is advantageously adapted to process a pre-existing family history database file containing family history information (such a file also variously referred to herein as "DB File"). In addition to including information about the interrelations among members of the family, the database file may also include personal information for each individual included in the family history. Besides the full name and birth date of each individual, the personal history for each individual could include information selected from the group including: the date and time of death,

marriages, parents, children, and personal attributes such as eye color, hair color, hair thickness, height, weight and various other attributes that could describe a person over their lifetime, such as a graphic image. Each person named in the DB File is preferably represented by what is referred to herein as a "node" or "data record" within the DB File. Stated differently, the DB File comprises a plurality of nodes or data records, each corresponding to a different individual. Each node or data record contains possibly multiple pieces of information about the corresponding individual. Collectively, the information about an individual that may be included in each node of the database file is referred to herein as "family information."

The present invention relates in one aspect to creating a so-called tree matrix comprising a plurality of tree matrix nodes, each corresponding to a node in the database file. Further, the invention relates to establishing "links" or "associations" between pairs of nodes in the tree matrix, where each link or association represents a familial relationship of some sort between the persons to whom the two associated tree matrix nodes correspond.

The family history database file can be established using various conventional methods. For example, a user may input, via a keyboard, the above family history information using a desired software package as the interface mechanism for the CPU storage medium. A software package such as Family Tree Maker®, manufactured and owned by The Learning Company Properties, Inc. of Cambridge, Massachusetts, is an example of such an interface for arranging the database. However, the invention is not limited as other types of genealogy, database or flow chart programs might also be advantageously employed. Similarly, other manufacturers may make these other programs available. It is believed that those of ordinary skill in the art having the benefit of the present disclosure will readily appreciate how the present invention may be advantageously applied in conjunction with various different family history database files, presently known or to be developed. For the purposes of the present disclosure, it is sufficient to characterize the family history database file generally as comprising an electronic (e.g., computer) file having data therein arranged and organized in such a way as to permit retrieval of family information about specified individuals. Although the present disclosure speaks in terms of "nodes" of information, those of ordinary skill in the art will appreciate that this characterization reflects the manner in which information is accessible from the database, and further that the internal structure and organization of such a database may differ widely from implementation to implementation.

In the presently disclosed embodiment, the database receives the family history information input by the user and organizes it into a conventional fixed database file structure using non-volatile memory. For example, a database file composed of multiple blocks or nodes of memory or data storage, where related information of each node is linked via pointers to other nodes within the database file. Those of ordinary skill in the art having the benefit of the present disclosure will appreciate that other types of conventional database files might also be employed.

In the presently preferred embodiment, the All-In-One application is implemented as one or more computer programs or modules written in the C++ programming language, although those of ordinary skill in the field of computer science will appreciate that the application(s) may be written in other programming languages.

As noted above, the invention may be implemented in part by programming a general-purpose processor. The programming may be accomplished through the use of a program storage device readable by the processor that encodes a program of instructions executable by the processor for performing the operations described above. The program storage device may take the form of, e.g., a floppy disk; a CD-ROM; a memory device (e.g., RAM, ROM, EPROM, EEPROM, etc.); and other forms of the kind well-known in the art or subsequently developed. The program of instructions may be "object code," i.e., in binary form that is executable more-or-less directly by the computer; in "source code" that requires compilation or interpretation before execution; or in some intermediate form such as partially compiled code. The program storage device may be one that is directly readable by the processor, or it may be one that is unusable by the processor per se but that provides intermediate storage of the program of instructions. The program of instructions may be read directly from the program storage device by the processor. The precise forms of the program storage device and of the encoding of instructions are immaterial here.

Process Flow Charts

The basic pseudo code or flow chart of Figure 3 illustrates the inventive process for traversing the pre-existing database file described above. In particular, after a user inputs a primary individual's name to define the family history file that will be organized, the "All-In-One" application begins to build the desired family tree matrix file or tree matrix. As illustrated, this functional process involves traversing the pre-existing

database file to "Expand" 40 the family history around the primary individual.

As will be described in more detail below for a specific embodiment, main Expand module 40 is recursive and includes various sub-modules of "BuildMatrix" 44, "GetParentsInfo" 50, "GetSpousesInfo" 60, and "GetChildrenInfo" 80. Sub-modules "GetParentsInfo" 50, "GetSpousesInfo" 60, and "GetChildrenInfo" 80 are also recursive. Together, these modules cooperate to derive, from the family history database file, a tree matrix file in which the genealogical associations among each of the individuals named in the database file are identified. The tree matrix can thus be used to create a "family tree," i.e., a graphical diagram of interrelationships among members of a family. The concept of a "family tree" is well-known to most people.

Expand (individual)

As illustrated by Figure 3, the Expand module of the All-In-One application provides controlling instructions to link the other sub-modules. With each module and sub-module, questions are presented that can be resolved by evaluating the information contained by the memory node for the individual of interest contained by the family history database file. This evaluation or scanning process is preferably conducted by a conventional method. For example, the scanning method can include the recognition of a specific flag or individual name within the node associated to the desired individual. Other terms commonly used to represent a flag within a software program would include the terms mark and tag.

The term "Expand," as used herein, describes a recursive method for locating and scanning a node corresponding to a desired individual named in the database file. More specifically, the locating step of Expand comprises the process of traversing the family history database file to find the node corresponding to the desired individual. Once found, that node is scanned for specific information that can be used to identify relationships between the individual and other persons represented by nodes in the database file. As such relationships are identified, a tree matrix is created to document such relationships. In one embodiment, the tree matrix itself is comprised of a plurality of nodes each corresponding to individuals represented in the database file. Additionally, the tree matrix defines associations or links between its nodes corresponding to the familial relationships identified during the "Expanding" operations in accordance with the presently disclosed embodiment. Such associations may correspond, for example, to a direct link to a parent (a parental association), marriage

(a spousal association), or child (an offspring association) of the desired individual.

Those of ordinary skill in the art will appreciate that the tree matrix in accordance with the presently disclosed embodiment of the invention may take the form of an electronic (e.g., computer) file comprising a plurality of data items and associated flags, pointers, links and the like. A node in the tree matrix is "created" when data about an individual is entered into the tree matrix file; familial relationships are represented by pointers or other links or references (associations) between and among nodes in the tree matrix. In one embodiment, the associations between nodes in the tree matrix may be differentiated according to the type of relationship to which they correspond (e.g., a spousal association may be differentiated from a parental association, and so on). It is believed that those of ordinary skill in the computer programming and database will readily appreciate how associations between data items in data structures such as the tree matrix contemplated herein may be defined, using pointers or other well-known data constructs. In addition, the skilled artisan should appreciate that the database file for the tree matrix may be fixed or temporary, e.g., maintained in non-volatile or volatile memory.

Because Expand module 40 is recursive, once an association is recognized and defined as an association between two nodes in the tree matrix, Expand module 40 executes again to identify any associations between the new individual and other persons represented by nodes in the database file. Expand module 40 terminates when the scanning step cannot establish an association to the individual being scanned. At such time, the All-In-One application will be complete. However, because a family history file may contain individuals that may not or cannot be directly associated to another individual of the pre-existing database file at present execution, the user may pre-set the All-In-One application to find all unrelated individuals that do not have a direct association. Consequently, as will be described in more detail below with reference to Figure 4F, Expand module 40 may be initiated again if the answer to a final question of "IS THERE AN ID REMAINING IN DB FILE?", is yes.

Figure 4A illustrates the functional steps of Expand module 40. In particular, once the node corresponding to a primary individual or individual of interest (hereinafter "ID node") is found within the pre-existing database file, the ID node will be scanned to determine if the "ID IS IN TREE MATRIX YET?", as represented by block 43 in Figure 4A. If the node for the ID indicates that the individual is not in the tree matrix, the sub-module "BuildMatrix" 44 is called; otherwise the same ID node will be scanned to

resolve if "ID EXPANDED?" 45. If the node for the individual is expanded, Expand module 40 is complete. On the other hand, if the node for the ID indicates that it has not been expanded yet, the sub-module "GetParentsInfo" 50 is called to evaluate the ID.

Once the node for the ID of interest has been evaluated in GetParentsInfo sub-module 50, the same ID node will be evaluated by GetSpousesInfo sub-module 60. Both of these sub-modules as well as BuildMatrix sub-module 44 will be described in further detail below.

Before the ID node can be evaluated by "GetChildrenInfo" sub-module 80, the ID node is scanned to resolve if "MARRIAGE EXPANDED?" as represented by block 70. Consequently, if the ID node provides a flag indicating that the marriage for the individual has been expanded, than the expansion for the ID node being considered will be done. However, if the marriage for the individual has not been expanded, the GetChildrenInfo sub-module 80 will evaluate the instant ID node.

The above process is repeated for all individuals named in the family history database file until all ancestors, descendants, multiple spouses, step parents, step children and in-laws having a direct link to the primary individual have been identified and the corresponding associations between nodes in the tree matrix are established. As described earlier, a direct link would include a situation where the node of the ID provides pointers to related individuals such as siblings, parents, spouses, grandparents, great-grandparents, and children.

BuildMatrix (individual)

Referring now to Figure 4B, the functional steps of BuildMatrix sub-module 44 are illustrated. As mentioned earlier, this sub-module is called by Expand module 40 when an ID node of the pre-existing family database file has not been entered into the tree matrix. Consequently, the function of this sub-module is to "CREATE A MATRIX NODE FOR ID", as represented by block 46, and pose the question "ID IN TREE MATRIX?" as represented by block 47.

If the individual is in the tree matrix, the last step in BuildMatrix sub-module 44 is to "MARK ID NODE OF DATABASE AS BEING IN TREE MATRIX", as represented by block 48. Otherwise, the last step is to "MARK ID NODE OF DATABASE AS BEING DUPLICATED IN TREE MATRIX", as represented by block 49. As indicated, after either of these steps, the ID node is passed back to Expand module 40 to be scanned

and determine if the "ID EXPANDED?", as represented by block 45.

The tree matrix is a database file of nodes. Each matrix node is linked to or associated with another matrix node to define a memory array structure. This matrix database structure is different from the family history database file. In addition, each matrix node will typically only contain a minimum amount of information about the ID along with stored address information, or pointers, to find the related database node within the database file. For example, an individual node of the database file may include information such as the date and time of death, marriages, parents, children, and personal attributes such as eye color, hair color, hair thickness, height, weight and various other attributes that could describe a person over their lifetime, such as a graphic image. However, an individual matrix node will typically only include the individual's name and date of birth and death. The information stored by any matrix node is preferably selectable by the user before the all-in-one family tree is created.

GetParentsInfo

Referring now to Figure 4C, the functional steps of GetParentsInfo sub-module 50 are illustrated. As mentioned above, sub-module 50 is called by Expand module 40 when an ID node of the pre-existing family history database file has been entered into the tree matrix. In turn, this same sub-module may be called after the BuildMatrix sub-module 44.

The first functional step of sub-module 50 determines if the ID node has any parents. If the ID node does not have any parents, the ID node is passed back into Expand module 40 (FIG. 4A) to be received by GetSpousesInfo module 60 (to be discussed in further detail below). However, if the ID node contains parent information, "FOR EACH SET OF PARENTS", as represented by block 51, GetParentsInfo sub-module 50 starts a recursive loop in which the first step includes "GET PARENT'S MARRIAGES FROM DATABASE FILE", as represented by block 52.

If "FATHER EXISTS?", as represented by block 53, the father ID node is retrieved from the family history database file by the "GET FATHER FROM DB FILE" step represented by block 54, and passed to the Expand module 40 for analysis. Otherwise, the ID node is scanned to identify if "MOTHER EXISTS?", as represented by block 55 or if a null ID node should be passed to the Expand module 40 for analysis. Similar to the situation when the father exits, if the mother exists, the mother ID node is retrieved from the database file by the "GET MOTHER FORM DB FILE" step,

represented by block 56, and passed to the Expand 40 module for analysis.

The foregoing process is conducted for each parent of an individual passed into the GetParentsInfo sub-module 50. If for some reason the information for father and/or mother does not exist, but the ID node indicates that it should, a null ID node is received by Expand module 40. This situation is typically present when an ID node indicates a sibling, but does not indicate a parent for the ID node. As will be discussed in more detail with reference to an example, the null ID node will be passed through the BuildMatrix sub-module like any other ID node to create the necessary positioning of all individuals of the family database file within the All-In-One matrix file.

GetSpousesInfo (individual)

Figure 4D illustrates high-level pseudo code for "GetSpousesInfo" sub-module 60. As mentioned above, this sub-module is called by the Expand module 40 after passing out of "GetParentsInfo" sub-module 50.

The first functional step of GetSpousesInfo sub-module 60 determines if the ID node of interest has any marriages or spouses. If the ID node does not have any marriages, the ID node is passed back into the Expand module 40 (FIG. 4A) to resolve the question, "MARRIAGE EXPANDED?", as represented by block 70. However, if the ID node contains marriage information, the GetSpousesInfo sub-module 50 performs a recursive loop "FOR EACH MARRIAGE", as represented by block 61.

With a marriage identified from the ID node, the GetSpousesInfo sub-module 60 will "GET MARRIAGE ID FROM DATABASE FILE", as represented by block 62 and "MARK ID NODE OF DB FILE AS EXPANDED" as represented by block 63. Next, the question of whether the "SPOUSE EXISTS?" represented by block 65 is resolved by scanning the database file for the identified spouse ID node. Provided that the step of "GET SPOUSE FROM DB FILE" as represented by block 66 is successful, the retrieved spouse ID node will be passed into Expand module 40 for evaluation and placement by BuildMatrix sub-module 44 into the tree matrix. However, if the spouse ID node cannot be found in the database file, the ID node passed into GetSpousesInfo sub-module 60 will proceed back to Expand module 40 to resolve the question "MARRIAGE EXPANDED?" as represented by block 70. (See Fig. 4A.)

GetChildrenInfo (marriage)

Figure 4E illustrates the high-level pseudo code for "GetChildrenInfo" sub-module 80. As mentioned above, this sub-module is called by Expand module 40 after resolving the question "Marriage expanded?" as represented by block 70 and being passed out of "GetSpousesInfo" sub-module 50.

Provided that the ID node of the database file has not been earlier tagged as having its marriage expanded, the first functional step of this sub-module will be to "MARK MARRIAGE IN DB FILE AS EXPANDED" as represented by block 81. Next, the database file is traversed to find all children ID nodes of the database file associated to the ID node received by the GetChildrenInfo sub-module 80. Consequently, "FOR EACH CHILD OF MARRIAGE" as represented by block 82, the instant sub-module will "GET CHILD FROM DB FILE" as represented by block 85 and pass the instant child ID node into the Expand module 40 for analysis.

GetRemainingID

Turning now to Figure 4F, the last step of the high-level pseudo code for Expand module 40 will be described. As noted above, this step is called by Expand module 40 after an ID of interest has been scanned to find that no associations have been expanded. In other words, all ancestors, descendant, in-laws, multiple spouses, and step-children have been created in the All-In-One tree matrix by traversing the database file to resolve all steps of Expand module 40 and sub-modules BuildMatrix 44, GetParentsInfo 50, GetSpousesInfo 60 and GetChildrenInfo 80. Consequently, this step acts as a final check to obtain all ID nodes of the pre-existing database file and position them in the tree matrix. Because any resulting ID node cannot be associated to the now existing tree matrix, the resulting ID nodes will not display or provide a link to the other members of the existing tree matrix.

To obtain the remaining individuals of the database file, Expand module 40 asks the question "IS THERE AN ID REMAINING IN DB FILE", as represented by block 90. As before, the solution to this question is resolved by traversing the database file to "GET ID FROM DB FILE" that has not been marked by BuildMatrix sub-module 44. If all ID nodes of database file are marked, the program is complete and the All-In-One family tree matrix can be displayed using any conventional method. However, if any ID node of the database file is not marked, the process of Figure 3 is started over since the

next step will be to "GET ID FROM DB FILE" as represented by block 91 and pass the retrieved ID node into the Expand module 40. As indicated above, as long as the user indicated at the start of the inventive All-In-One application that all un-related individuals should be found, this last step of Expand module 40 will continue to be called until all ID nodes of the family history database file have been found and positioned within the desired family matrix tree.

The skilled artisan will recognize that the inventive process as described above with reference to Figures 4A - 4F, creates various temporary marks or flags for each ID database node as the All-In-One tree matrix file is created. In particular, each ID database node under consideration is marked to indicate a specific level of the process that has been completed, such as, "ID IN TREE MATRIX?", "ID EXPANDED?", and "MARRIAGE EXPANDED?" as represented by blocks 43, 45, and 70, respectively, of Expand module 40, Figure 4A. Once the process is complete (e.g., the All-In-One application is terminated, the desired All-In-One matrix file is created, or a new All-In-One matrix file is requested), the marks are removed from the ID nodes of the database file so that a future family tree matrix can be created, which may include new or different members of the family database file.

In addition, the skilled artisan will recognize that given the recursive nature of Expand module 40 and its recursive sub-modules, the process of linking each individual to define their association in the family tree matrix file can be defined as a complex tiering structure, where each tier may define a recursive step to be completed. For example, a tiering structure may develop that explores various sub-modules relative to a single ID to find their ancestors and descendant before falling back to the same sub-modules of Expand module 40 to explore the primary ID.

This fractal process style has been proven to be a very practical, efficient and effective way to organize and graphically illustrate any family history that a user may develop, no matter how disjointed the information. Consequently, persons of ordinary skill in software and database technologies will appreciate that the information of the pre-existing database file can be collected in any order using this style to form the desired tree matrix. For example, once a primary individual is identified, the main and sub-modules of the invention can be structured to collect any or all associations identified by the primary individual, and is not so limited to a specific order.

Having described the preferred system and high level pseudo code with reference to Figures 1, 2 and 4A-4F, an example of the process of implementation will

now follow. To initiate this process, it will be assumed that the user indicates the All-In-One Family Matrix tree should be started with ID4. Consequently, the steps according to the inventive pseudo code will proceed as follows.

Analysis of Conventional Family Tree Database File using Functional Flow Charts (Figures 4A-4F) to Develop Family Tree Matrix of Figures 5A-5H

Process Step	Functional Step	Individual ("ID") Expanded
1.	#40	ID4.
2.	#43	ID4
3.	#44	ID4
4.	#46	ID4
5.	#47	ID4
6.	#48	ID4
7.	#45	ID4
8.	#50	ID4
9.	#51	ID4 (no sets of parents)
10.	#60	ID4
11.	#61	(marriage between ID4 and ID1)
12.	#62	ID4
13.	#63	ID4
14.	#65	ID4
15.	#66	(get ID1)
16.	#40	ID1
17.	#43	ID1
18.	#44	ID1
19.	#46	ID1 (position relative to relationship)

At this stage of the process, Figure 5B illustrates the creation and positioning of tree matrix node ID1 relative to its association 100 with ID4. In this particular example, association 100 defines the marriage between ID4 and ID1.

20.	#47	ID1
21.	#48	ID1
22.	#45	ID1
23.	#50	ID1
24.	#51	ID1 (no sets of parents)
25.	#60	ID1
26.	#61	ID1
27.	#62	(marriage between ID4 and ID1)
28.	#63	ID1
29.	#65	ID1
30.	#66	(get ID4)
31.	#40	ID4
32.	#43	ID4
33.	#45	ID4 (already expanded)
34.	#70	ID1

35.	#80	ID1
36.	#81	ID1
37.	#82	(two children: ID5 and ID 6)
38.	#85	(get ID5)
39.	#40	ID5
40.	#43	ID5
41.	#44	ID5
42.	#46	ID5 (position relative to relation ID1)

At this stage of the process, Figure 5C illustrates the creation and positioning of matrix node ID5 relative to its association 102 with ID1. In this particular example, association 102 defines ID5 as a child of parents ID4 and ID1, and ID1 to be the mother of child ID5 because children are positioned below their parents.

43.	#47	ID5
44.	#48	ID5
45.	#45	ID5
46.	#50	ID5
47.	#51	ID5
48.	#52	(marriage between ID4 and ID1)
49.	#53	ID5
50.	#54	(get ID 4)
51.	#40	ID4
52.	#43	ID4
53.	#45	ID4 (already expanded)
54.	#60	ID5 (no spouses)
55.	#61	ID5
56.	#70	ID5
57.	#80	ID5 (no children)
58.	#81	ID5
59.	#85	(get ID 6)
60.	#40	ID6
61.	#43	ID6
62.	#44	ID6
63.	#46	ID6 (position relative to relation ID5)

At this stage of the process, Figure 5D illustrates the creation and positioning of tree matrix node ID6 relative to its association 104 with ID5. In this particular example, association 104 defines another child of parent ID4 and ID1. Similar to marriages, siblings are always positioned horizontally spaced from associated brothers and/or sisters.

64.	#47	ID6
65.	#48	ID6
66.	#45	ID6
67.	#50	ID6
68.	#51	ID6

	69.	#52	(marriage between ID4 and ID1)
	70.	#53	ID6
	71.	#54	(get ID 4)
	72.	#40	ID4
5	73.	#43	ID4
	74.	#45	ID4 (already expanded)
	75.	#60	ID6
	76.	#61	ID6
	77.	#62	(marriage between ID6 and ID17)
10	78.	#63	ID6
	79.	#65	ID6
	80.	#66	(get ID17)
	81.	#40	ID17
	82.	#43	ID17
15	83.	#44	ID17
	84.	#46	ID17 (position relative to relation ID6)

At this stage of the process, Figure 5E illustrates the creation and positioning of tree matrix node ID17 relative to its association 106 with ID6. In this particular example, association 106 defines the marriage between ID17 and ID6.

20	85.	#47	ID17
	86.	#48	ID17
	87.	#45	ID17
	88.	#50	ID17
	89.	#51	ID17
25	90.	#52	(marriage between ID19 and unknown spouse)
	91.	#53	ID17
	92.	#54	(get ID19)
	93.	#40	ID19
	94.	#43	ID19
30	95.	#44	ID19
	96.	#46	ID19 (position relative to relation ID17)

At this stage of the process, Figure 5F illustrates the creation and positioning of tree matrix node ID19 relative to its association 108 with ID17. In this particular example, association 108 defines parent(s) ID19 of ID17.

35	97.	#47	ID19
	98.	#48	ID19
	99.	#45	ID19
	100.	#50	ID19
	101.	#51	ID19 (no sets of parents)
40	102.	#60	ID19
	103.	#61	ID19
	104.	#62	(marriage between ID19 and unknown spouse)
	105.	#65	ID19 (no spouse)
	106.	#70	ID19
45	107.	#80	ID19

108.	#81	ID19
109.	#82	(two children: ID 17 and ID20)
110.	#85	(get ID17)
111.	#40	ID17
112.	#43	ID17
113.	#45	ID17
114.	#50	ID17
115.	#51	ID17
116.	#52	(marriage between ID19 and unknown spouse)
117.	#53	ID17
118.	#54	(get ID19)
119.	#40	ID19
120.	#43	ID19
121.	#45	ID19 (already expanded)
122.	#60	ID17
123.	#61	ID17
124.	#62	(marriage between ID6 and ID17)
125.	#65	ID17
126.	#66	(get ID6)
127.	#40	ID6
128.	#43	ID6
129.	#45	ID6 (already expanded)
130.	#70	ID17
131.	#80	ID17
132.	#81	ID17
133.	#82	(one child: ID 18)
134.	#85	(get ID 18)
135.	#40	ID18
136.	#43	ID18
137.	#44	ID18
138.	#46	ID18 (position relative to relation ID17)

At this stage of the process, Figure 5G illustrates the creation and positioning of matrix node ID18 relative to its association 110 with ID17. In this particular example, association 110 defines the children of parents ID6 and ID17.

139.	#47	ID18
140.	#48	ID18
141.	#45	ID18
142.	#50	ID18
143.	#51	ID18
144.	#52	(marriage between ID6 and ID17)
145.	#53	ID18
146.	#54	(get ID6)
147.	#40	ID6
148.	#43	ID6
149.	#45	ID6 (already expanded)
150.	#60	ID18 (no spouses)
151.	#61	ID18
152.	#70	ID18

153.	#80	ID18 (no children)
154.	#81	ID18
155.	#85	(get ID20)
156.	#40	ID20
157.	#43	ID20
158.	#44	ID20
159.	#46	ID20 (position relative to relation ID17)

At this stage of the process, Figure 5H illustrates the creation and positioning of tree matrix node ID20 relative to its association 112 with ID17. In this particular example, association 112 defines another child of parent ID19.

160.	#47	ID20
161.	#48	ID20
162.	#45	ID20
163.	#50	ID20
164.	#51	ID20
165.	#52	(marriage between ID19 and unknown spouse)
166.	#53	ID20
167.	#54	(get ID19)
168.	#40	ID19
169.	#43	ID19
170.	#45	ID19 (already expanded)
171.	#60	ID20 (no spouses)
172.	#61	ID20
173.	#70	ID20
174.	#80	ID20 (no children)
175.	#81	ID20
176.	#70	ID6 (marriage already expanded by ID17)
177.	#70	ID4 (marriage already expanded by ID1)

The above example finds all the ancestors, descendants, step-children and in-laws of the primary individual. No multiple marriages or unrelated individuals are found either because the database file does not contain such information, or the user defined such a limitation before the All-In-One application was initiated. Step-children will not be recognizable in the tree matrix by their association unless the step-children ID nodes provide information about multiple parents.

Alternative embodiment - multiple marriages

Analysis of Conventional Family Tree Database File using Functional Flow Charts (Figures 4A-4F) to Develop Family Tree Matrix of Figures 6A - 6C

This example, assumed to continue from above step 176, FIG. 5H, to explore a second marriage with reference to FIGS. 6A - 6C (note that in process step 10, ID4 now has two marriages: marriage between ID4 and ID1 and marriage between ID4 and ID7):

Process Step	Functional Step	Individual ("ID") Expanded
177.	#62	(marriage between ID4 and ID7)
178.	#63	ID4
179.	#65	ID4
180.	#66	(get ID7)
181.	#40	ID7
182.	#43	ID7
183.	#44	ID7
184.	#46	ID7 (position relative to relation ID4)
185.	#47	ID7
186.	#48	ID7
187.	#45	ID7
188.	#50	ID7
189.	#51	ID7
190.	#52	(marriage between ID8 and ID9)
191.	#53	ID7
192.	#54	(get ID8)
193.	#40	ID8
194.	#43	ID8
195.	#44	ID8
196.	#46	ID8 (position relative to relation ID7)
197.	#47	ID8
198.	#48	ID8
199.	#45	ID8
200.	#50	ID8
201.	#51	ID8 (no sets of parents)
202.	#60	ID8
203.	#61	ID8
204.	#62	(marriage between ID8 and ID9)
205.	#63	ID8
206.	#65	ID8
207.	#66	(get ID9)
208.	#40	ID9
209.	#43	ID9
210.	#44	ID9
211.	#46	ID9 (position relative to relation ID7)
212.	#47	ID9
213.	#48	ID9
214.	#45	ID9
215.	#50	ID9
216.	#51	ID9 (no sets of parents)
217.	#60	ID9
218.	#61	ID9
219.	#62	(marriage between ID8 and ID9)
220.	#63	ID9
221.	#65	ID9
222.	#66	(get ID8)
223.	#40	ID8
224.	#43	ID8

225.	#45	ID8 (already expanded)
226.	#70	ID9
227.	#80	ID9
228.	#81	ID9
229.	#82	(one child: ID7)
230.	#85	(get ID7)
231.	#40	ID7
232.	#43	ID7
233.	#45	ID7
234.	#50	ID7
235.	#51	ID7
236.	#52	(marriage between ID8 and ID9)
237.	#53	ID7
238.	#54	(get ID8)
239.	#40	ID8
240.	#43	ID8
241.	#45	ID8 (already expanded)
242.	#60	ID7
243.	#61	ID7
244.	#62	(marriage between ID4 and ID7)
245.	#63	ID7
246.	#65	ID7
247.	#66	(get ID4)
248.	#40	ID4
249.	#43	ID4
250.	#45	ID4 (already expanded)
251.	#70	ID7
252.	#80	ID7 (no children)
253.	#81	ID7
254.	#70	ID4 (marriage already expanded by ID1)
255.	#60	(everyone in tree)
256.	#61	DONE

Alternative embodiment - step children

Analysis of Conventional Family Tree Database File using Functional Flow Charts (Figures 4A-4F) to Develop Family Tree Matrix of Figures 7A - 7B

With reference to FIGS. 7A - 7B, continuing after above step 241, FIG. 6C, a second set of parents are explored and a stepchild association is defined. (Note that in step 235, ID7 now has two sets of parents: marriage between ID8 and ID9 and a marriage between an unknown couple):

Process Step	Functional Step	Individual ("ID") Expanded
242.	#52	(marriage of unknown couple)
243.	#53	ID7 (father doesn't exist)
244.	#25	ID7 (mother doesn't exist)
245.	#40	ID0 (place holder parent)
246.	#43	ID0

247.	#44	ID0
248.	#46	ID0 (position relative to relation ID7)
249.	#47	ID0
250.	#48	ID0
251.	#45	ID0
252.	#50	ID0
253.	#51	ID0 (no sets of parents)
254.	#60	ID0 (no spouses)
255.	#61	ID0
256.	#70	ID0
257.	#80	ID0
258.	#81	ID0
259.	#82	(two children: ID7 and ID10)
260.	#85	(get ID7)
261.	#40	ID7
262.	#43	ID7
263.	#45	ID7
264.	#50	ID7
265.	#51	ID7 (two sets of parents)
266.	#52	(marriage of ID8 and ID9)
267.	#53	ID7
268.	#54	(get ID8)
269.	#40	ID8
270.	#43	ID8
271.	#45	ID8 (already expanded)
272.	#52	(marriage of unknown couple)
273.	#53	ID7 (father doesn't exist)
274.	#25	ID7 (mother doesn't exist)
275.	#40	ID0 (place holder parent)
276.	#43	ID0
277.	#45	ID0 (already expanded)
278.	#60	ID7
279.	#61	ID7
280.	#62	(marriage between ID4 and ID7)
281.	#63	ID7
282.	#65	ID7
283.	#66	(get ID4)
284.	#40	ID4
285.	#43	ID4
286.	#45	ID4 (already expanded)
287.	#70	ID7
288.	#80	ID7 (no children)
289.	#81	ID7
290.	#82	(get ID10)
291.	#40	ID10
292.	#43	ID10
293.	#44	ID10
294.	#46	ID10 (position relative to relation ID7)
295.	#47	ID10
296.	#48	ID10
297.	#45	ID10

298.	#50	ID10
299.	#51	ID10
300.	#52	(marriage of unknown couple)
301.	#53	ID10 (father doesn't exist)
302.	#25	ID10 (mother doesn't exist)
303.	#40	ID0 (same place holder parent)
304.	#43	ID0
305.	#45	ID0 (already expanded)
306.	#60	ID10 (no spouses)
307.	#61	ID10
308.	#70	ID10
309.	#80	ID10 (no children)
310.	#81	#70, ID4 (marriage already expanded by ID1)
311.	#60	(everyone in tree)

Alternative Embodiment - ID with unknown relation

Analysis of Conventional Family Tree Database File using Functional Flow Charts (Figures 4A-4F) to Develop Family Tree Matrix of Figure 8

More than likely, the family history file will include individuals that cannot be directly linked to all members of the family history. In other words, an individual of the family history file will not be associated to another member of the file because the linking information, or research to define the association, has not been established. Since the user will typically want these individuals included in the family tree so that the gaps can be easily recognized, it is desirable for the current "All-In-One" application to accommodate this problem. Consequently, the following example illustrated in Figure 8 will address a solution to this problem.

With this final example, continuing after above step 292, FIG. 7B, an unrelated individual 22 having a node in the database file is expanded:

Process Step	Functional Step	Individual ("ID") Expanded
293.	#60	(we ID22 not in tree)
294.	#61	(get ID22)
295.	#40	ID22
296.	#43	ID22
297.	#44	ID22
298.	#46	ID22 (position relative to relation ID18)
299.	#47	ID22
300.	#48	ID22
301.	#45	ID22
302.	#50	ID22 (no sets of parents)
303.	#60	ID22 (no spouses)
304.	#61	ID22
305.	#70	ID22
306.	#80	ID22 (no children)
307.	#81	ID22

308.	#60	(everyone in tree)
309.	#61	DONE

In summary, a conventional family tree software program can only display and organize the descendants or ancestors of any family history file. This limitation prevents users from being able to visually inspect an entire database file of information related to a family history. The inventive process will allow a user to organize a conventional family tree database file to a matrix database file or matrix tree file. The matrix tree file allows both ancestors and descendants of a family history database file to be simultaneously displayed. In addition, related family history of in-laws, step-children, multiple spouses, and unrelated individuals that can not be clearly linked to the family history can also be organized for display if desired. Consequently, the present invention provides a reliable and effective way to display all information related to a family history.

From the foregoing detailed description of a specific embodiment of the invention, it should be apparent that a method of processing a family history file so as to derive therefrom a matrix tree file in which the interrelationships among members of the family whose historical information is contained in the family history file are identified. Although a specific embodiment of the invention has been described herein in some detail, it is to be understood that this has been done solely for the purposes of illustrating various features and aspects of the invention, and is not intended to be limiting with respect to the scope of the invention. Those of ordinary skill in the art having the benefit of the present disclosure will appreciate that the invention described herein is susceptible to various alternative implementations and design philosophies. For example, once a primary individual is found, Expand module 40 could be structured to obtain a spouse and/or child association(s) before obtaining the parent association(s) for any or all associations of the primary individual. Subsequently, it is contemplated that various substitutions, alterations, and/or modifications, including but not limited to those which may have been specifically noted in this disclosure, may be made to the disclosed embodiment without departing from the spirit and scope of the invention as defined in the appended claims, which follow.

CLAIMS:

1. A method of deriving a family tree matrix from a database file containing a plurality of nodes wherein each node contains family information about and individual, said method comprising:

(a) selecting a node in said database file corresponding to a primary individual;

(b) creating a tree matrix node corresponding to said primary individual;

(c) identifying any nodes in said database file corresponding to parents of said primary individual;

(d) for each node identified in step (c), creating a tree matrix node corresponding to the individual corresponding to said identified node;

(e) defining in said tree matrix a parental association between each tree matrix node created in step (d) and said tree matrix node corresponding to said primary individual;

(f) identifying any nodes in said database file corresponding to spouses of said primary individual;

(g) for each node identified in step (f), creating a tree matrix node corresponding to the individual corresponding to said identified node;

(h) defining in said tree matrix a spousal association between each tree matrix node created in step (g) and said tree matrix node corresponding to said primary individual;

(i) identifying any nodes in said database file corresponding to children of said primary individual;

(j) for each node identified in step (i), creating a tree matrix node corresponding to the individual corresponding to said identified node;

(k) defining in said tree matrix an offspring association between each tree matrix node created in step (j) and said tree matrix node corresponding to said primary individual; and

(l) successively selecting each remaining node in said database file to be said primary individual and repeating steps (b) through (k) for each said remaining node in said database file.

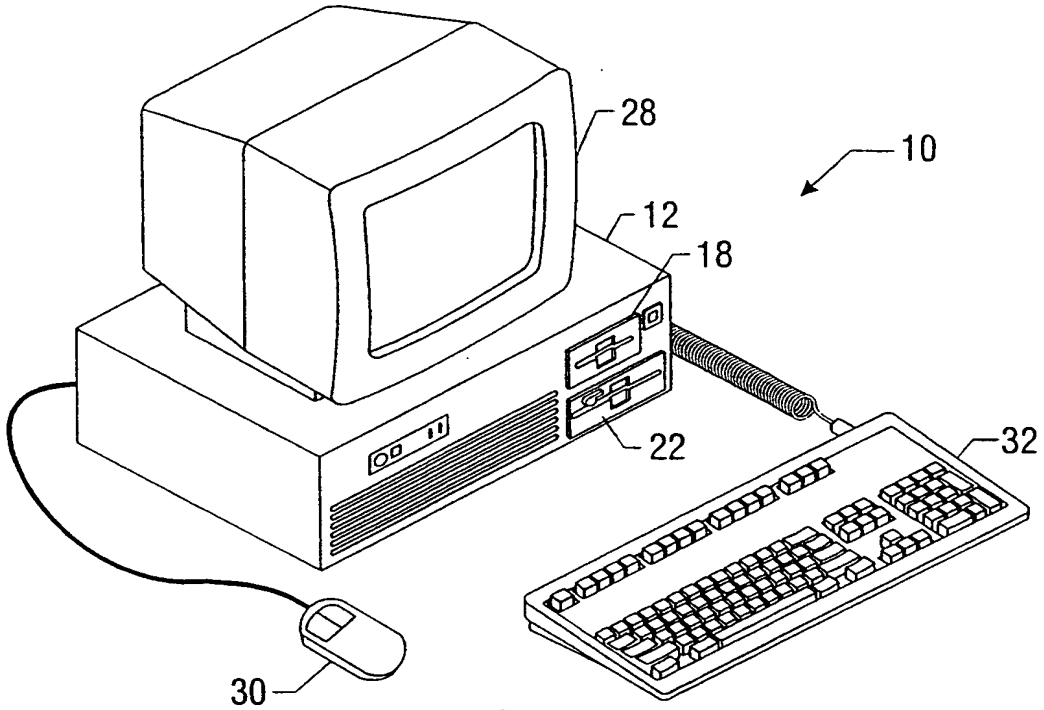


FIG. 1

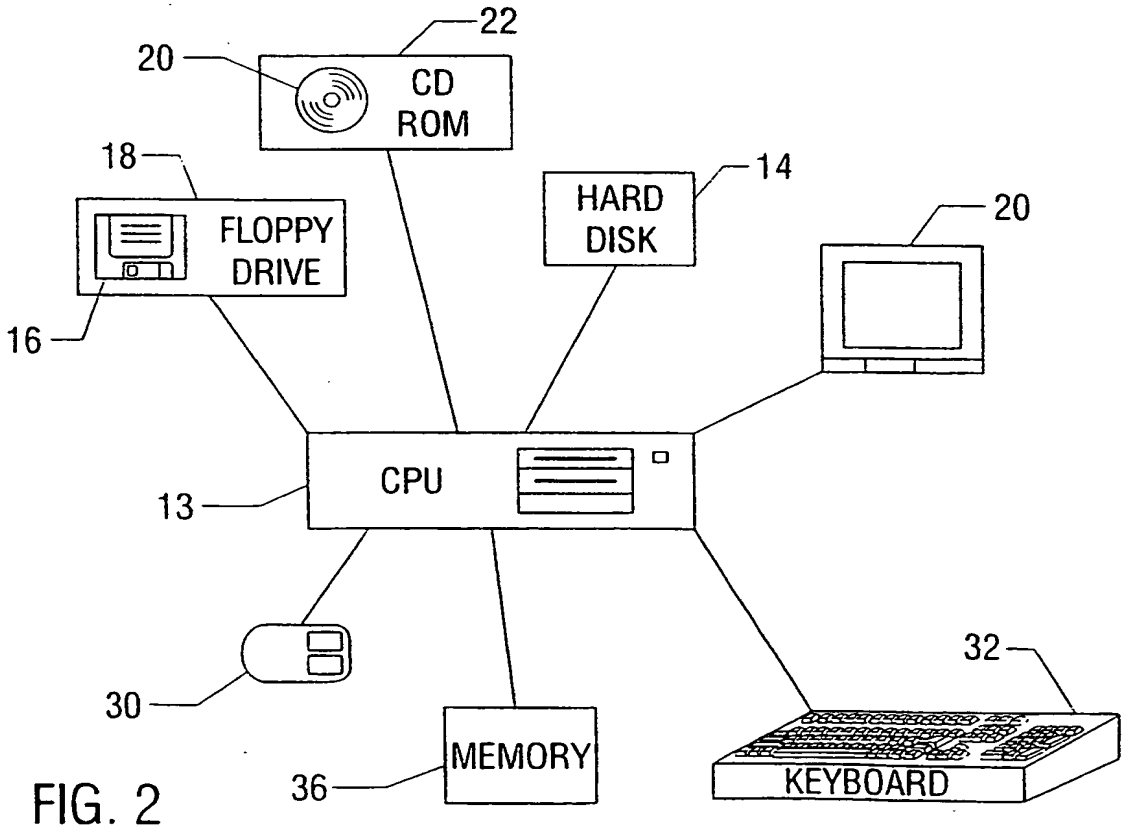


FIG. 2

BNSDOCID: <WO_____0057303A1_I_>

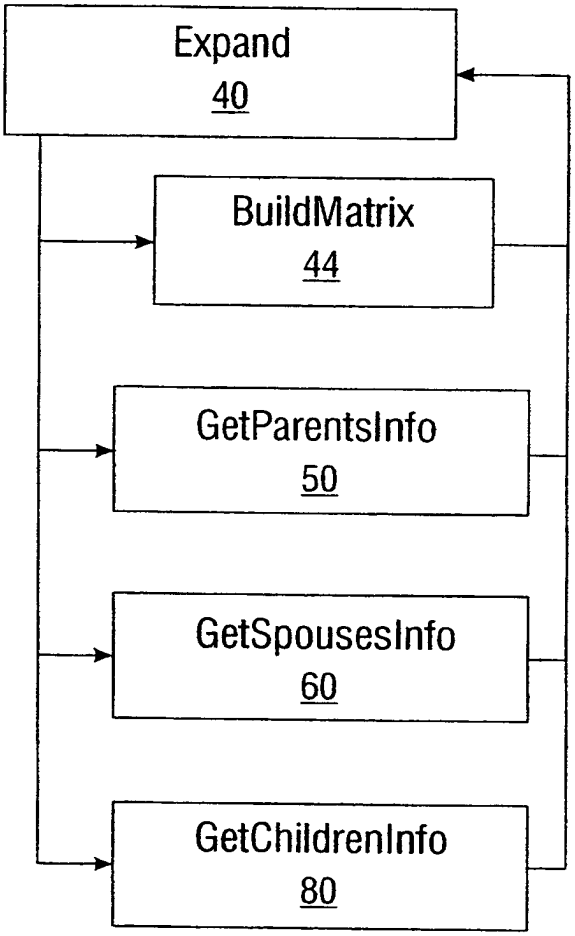


FIG. 3

BNSDOCID: <WO_____0057303A1_I_>

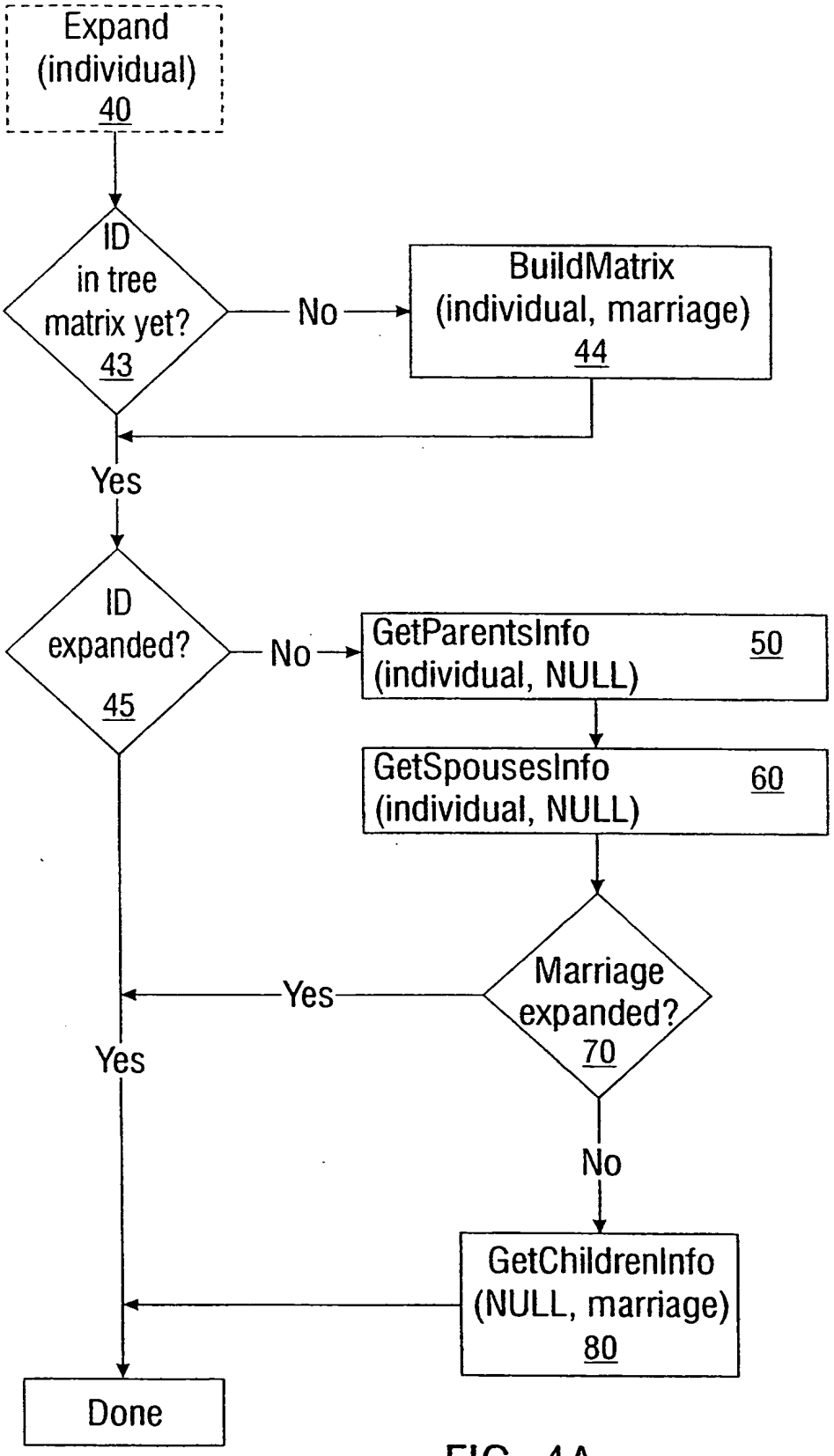


FIG. 4A

BNSDOCID: <WO_____0057303A1_I_>

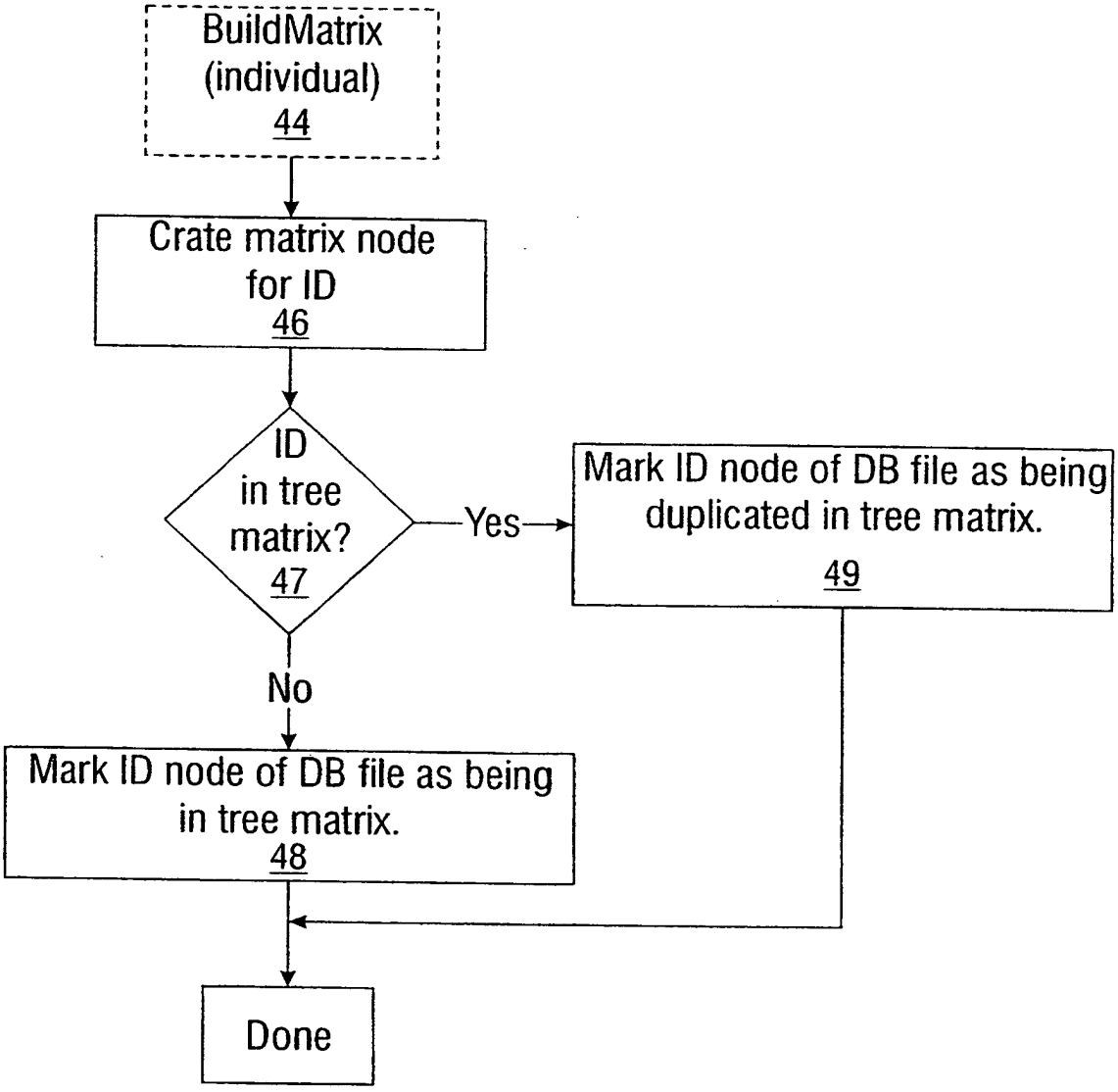


FIG. 4B

BNSDOCID: <WO_____0057303A1_I_>

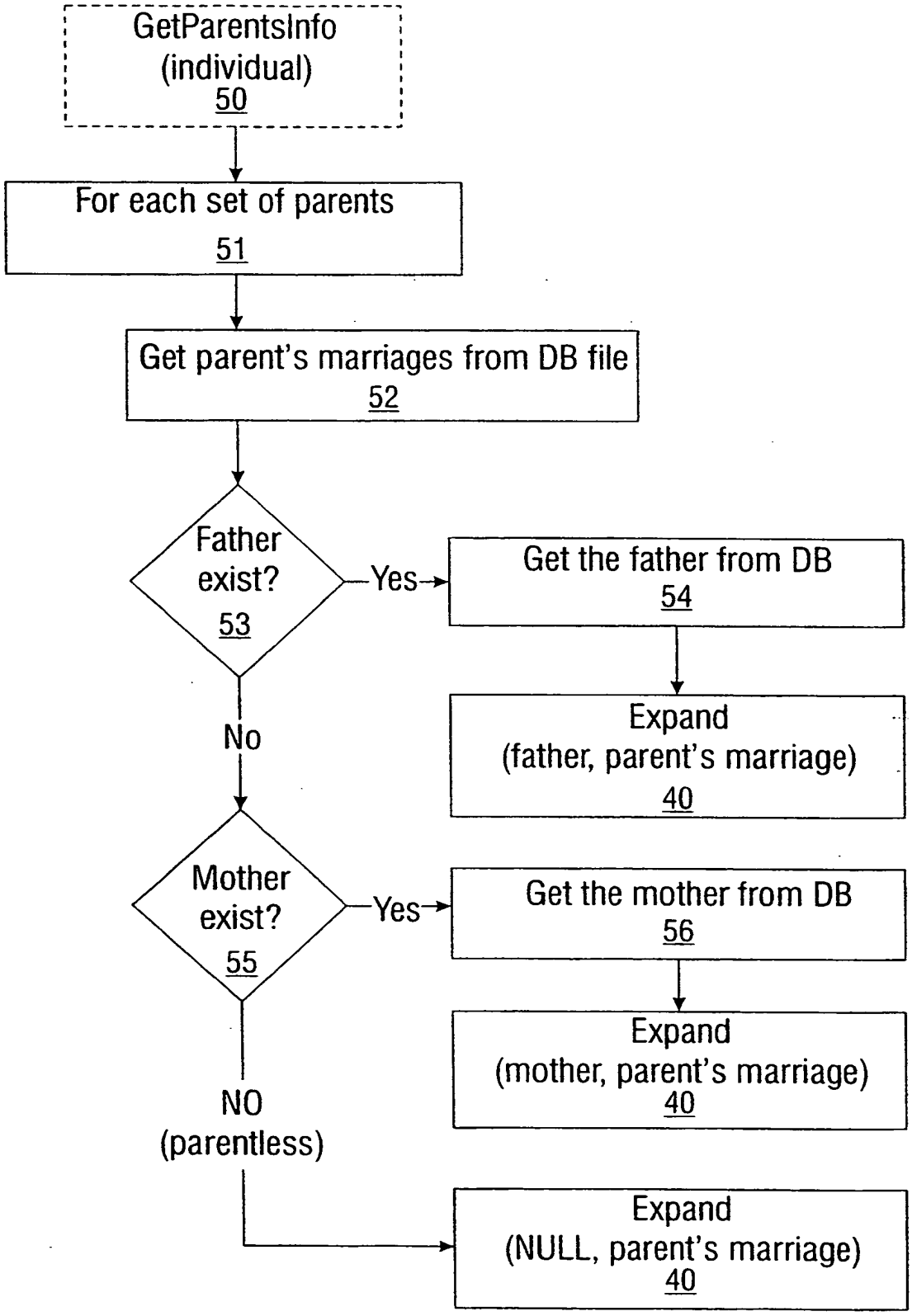


FIG. 4C

BNSDOCID: <WO_____0057303A1_I_>

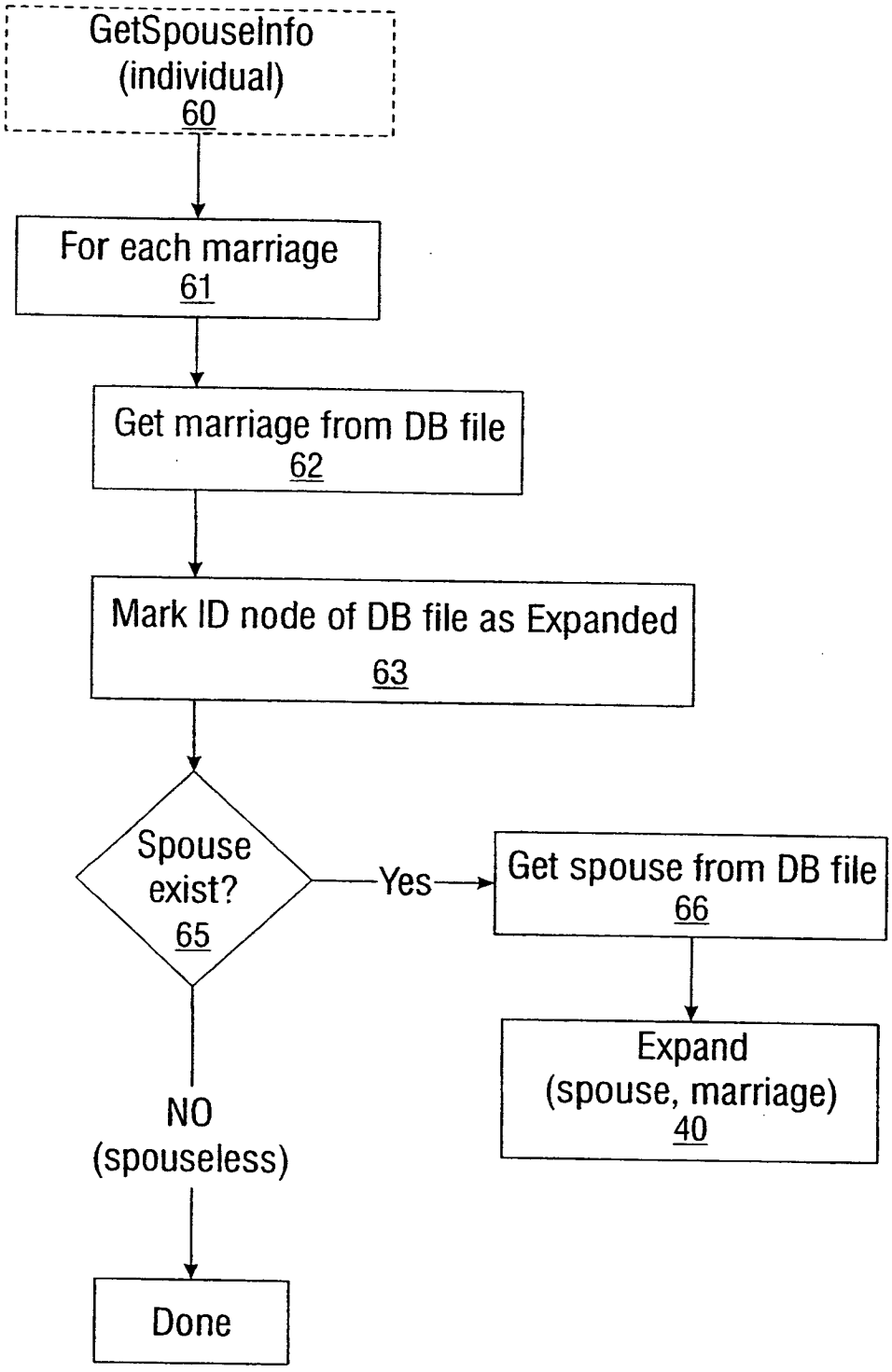


FIG. 4D

BNSDOCID: <WO_____0057303A1_I_>

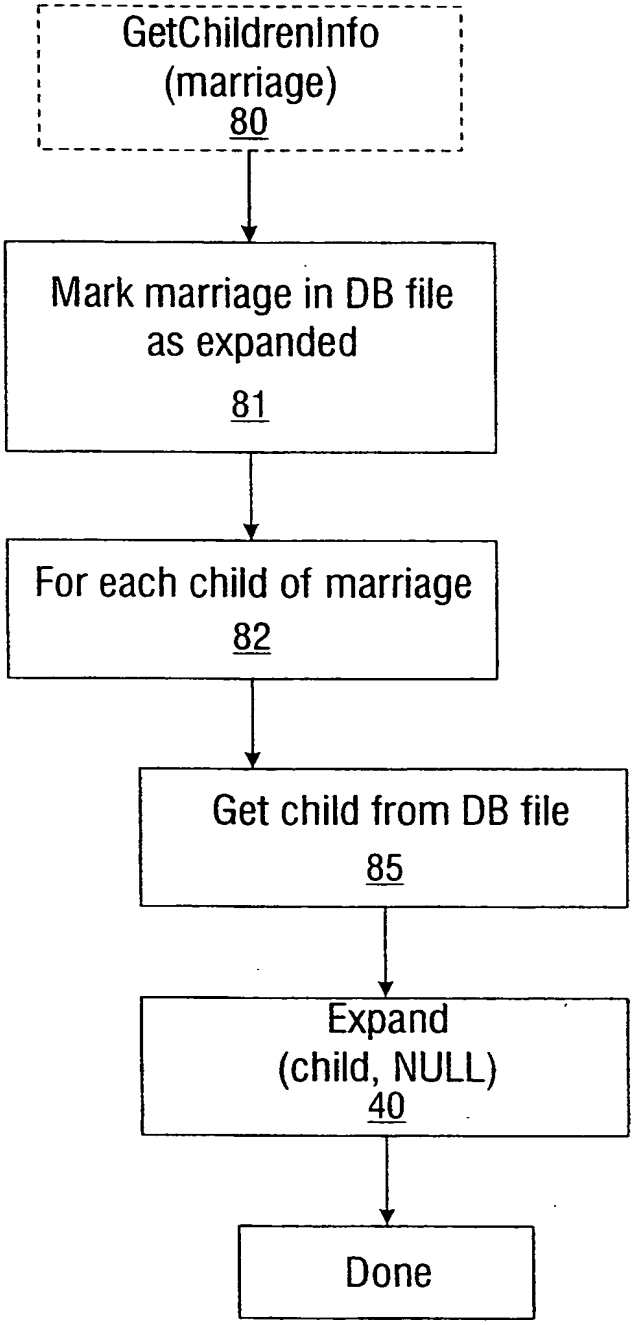


FIG. 4E

BNSDOCID: <WO_____0057303A1_I_>

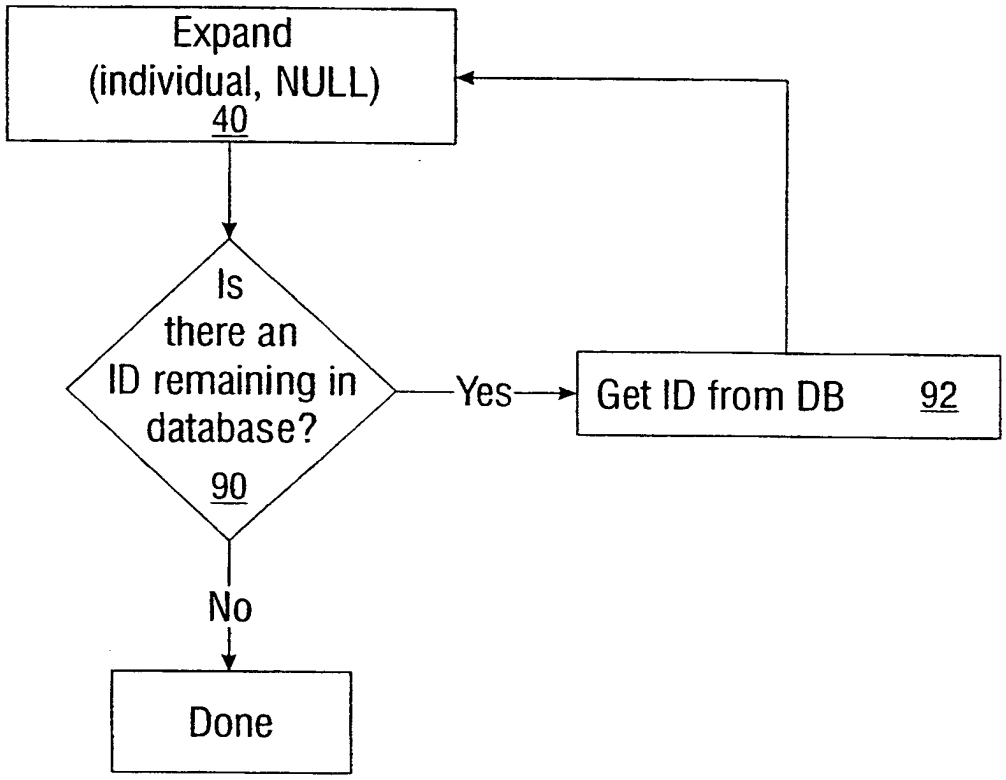


FIG. 4F

BNSDOCID: <WO_____0057303A1_I_>

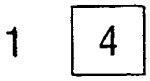


FIG. 5A

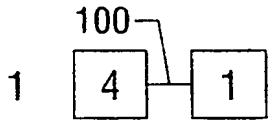


FIG. 5B

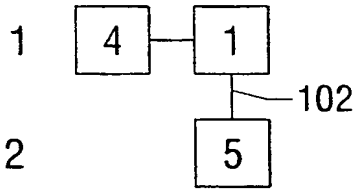


FIG. 5C

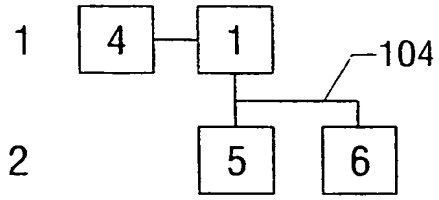


FIG. 5D

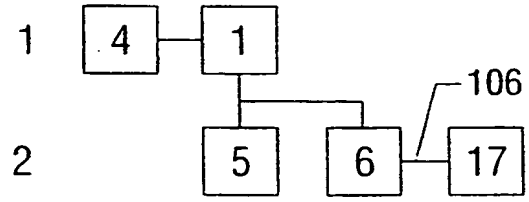


FIG. 5E

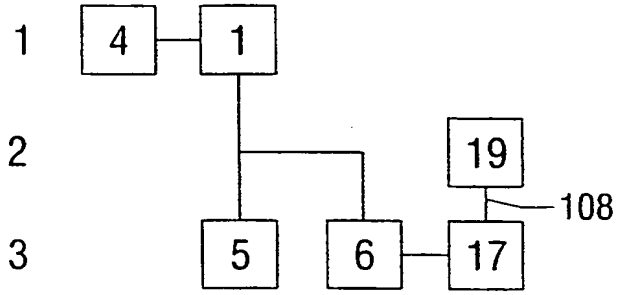


FIG. 5F

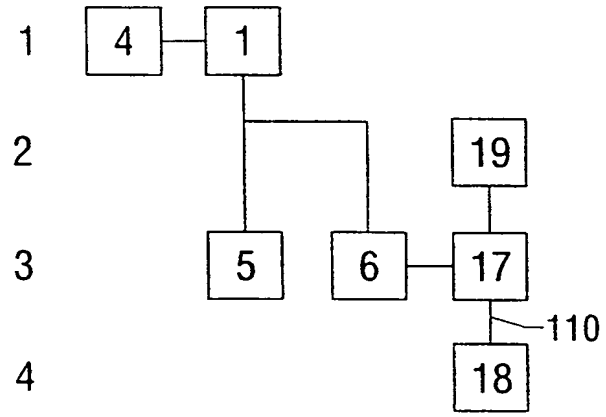


FIG. 5G

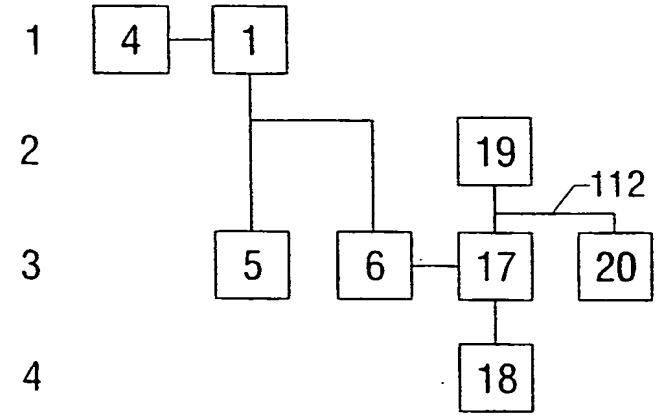


FIG. 5H

BNSDOCID: <WO_____0057303A1_I_>

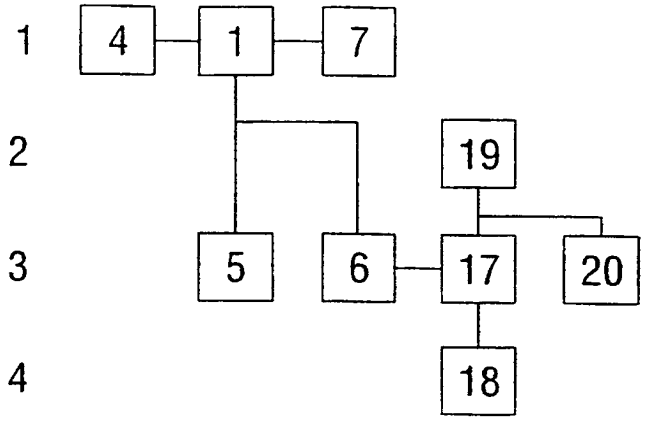


FIG. 6A

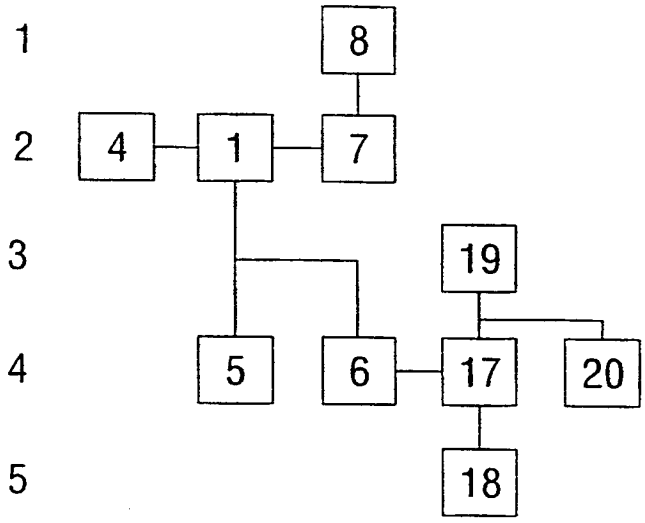


FIG. 6B

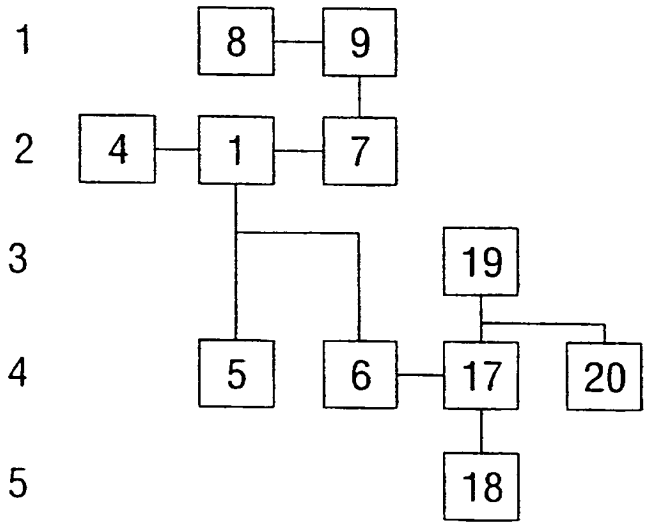


FIG. 6C

BNSDOCID: <WO_____0057303A1_I_>

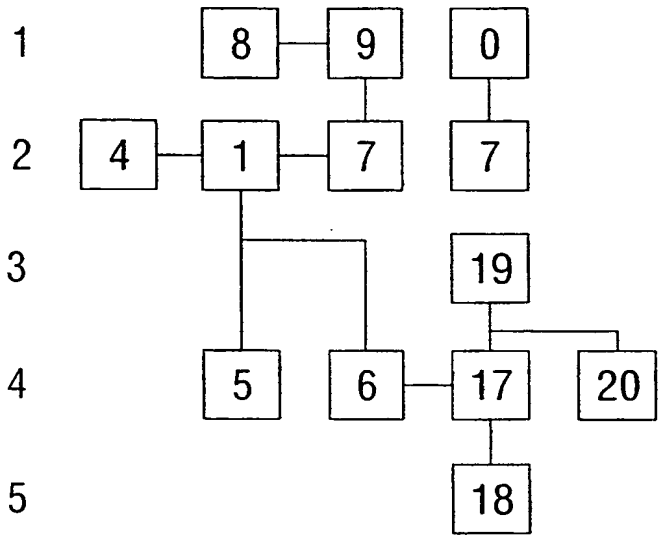


FIG. 7A

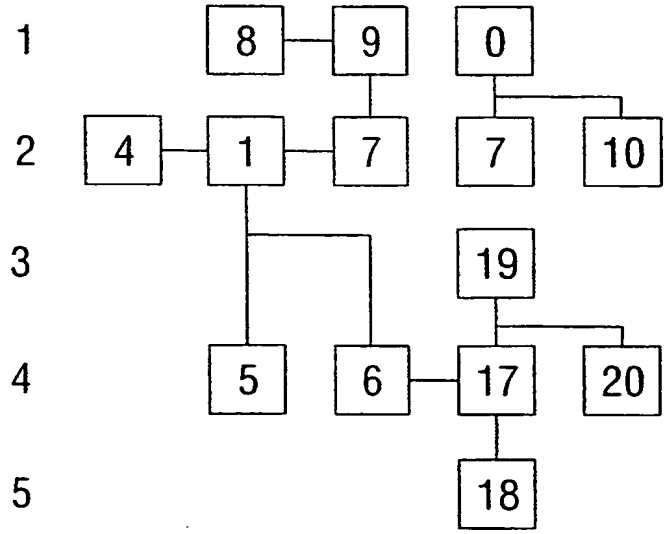


FIG. 7B

BNSDOCID: <WO_____0057303A1_I_>

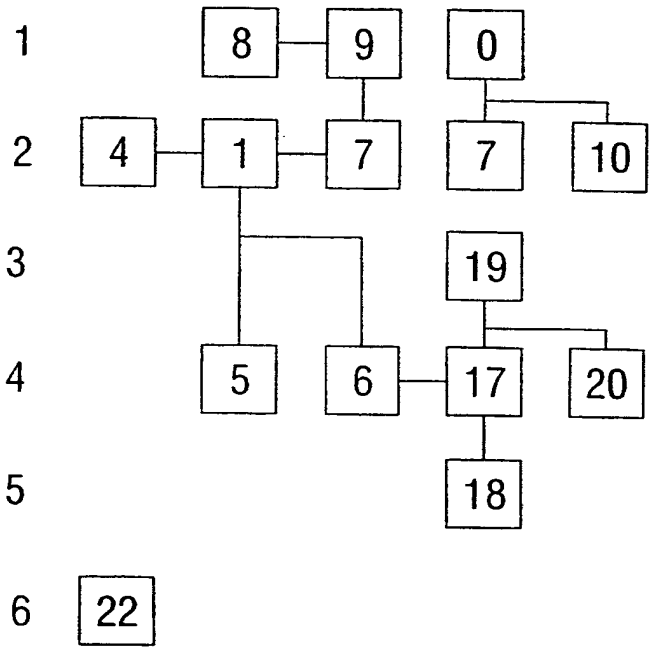


FIG. 8

BNSDOCID: <WO_____0057303A1_I_>

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US00/06331

A. CLASSIFICATION OF SUBJECT MATTER

IPC(7) : G06F 17/30
US CL : 707/100, 102; 705/27; 345/119, 340, 440
According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

U.S. : 707/100, 102; 705/27; 345/119, 340, 440

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

WEST, EAST
Search terms: nodes, hierarchical, hierarchy, tree, expand, database record, database file

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	US 5,644,740 A (KIUCHI) 01 July 1997, col. 1, lines 9-34 and lines 46-67, col. 2, lines 1-67, col. 3, lines 1-3 and lines 28-64, col. 4, lines 14-67, col. 5, lines 1-14, col. 7, lines 47-67, col. 8, lines 1-67, col. 11, lines 11-67, col. 12, lines 1-67, col. 13, lines 1-67, col. 14, lines 1-67, col. 15, lines 1-67, col. 16, lines 1-67, col. 17, lines 6-17 and lines 22-44, col. 18, lines 17-31 and 55-65, col. 19, lines 18-67, col. 20, lines 1-67, col. 21, lines 1-8 and 17-65, and col. 1-63.	1

[X] Further documents are listed in the continuation of Box C. [ ] See patent family annex.

*	Special categories of cited documents:	"T"	later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"A"	document defining the general state of the art which is not considered to be of particular relevance	"X"	document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"E"	earlier document published on or after the international filing date	"Y"	document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"L"	document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)	"&"	document member of the same patent family
"O"	document referring to an oral disclosure, use, exhibition or other means		
"P"	document published prior to the international filing date but later than the priority date claimed		

Date of the actual completion of the international search 19 MAY 2000	Date of mailing of the international search report 19 JUL 2000
Name and mailing address of the ISA/US Commissioner of Patents and Trademarks Box PCT Washington, D.C. 20231 Facsimile No. (703) 305-3230	Authorized officer KIM VU Telephone No. (703) 305-4393

Form PCT/ISA/210 (second sheet) (July 1998)*

BNSDOCID: <WO_____0057303A1_I_>

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US00/06331

C (Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	US 5,701,137 A (KIERNAN et al) 23 December 1997, col. 1, lines 25-67, col. 2, lines 1-3 and lines 15-48, col. 3, lines 2-23, col. 5, lines 50-63, col. 6, lines 7-67, col. 7, lines 1-67, col. 8, lines 1-67, col. 9, lines 1-25 and 61-67, col. 10, lines 1-67, col. 11, lines 1-44 and 59-67, and col. 12, lines 1-13, and col. 14, lines 3-15.	1
Y	US 5,606,669 A (BERTIN et al) 25 February 1997, col. 5, lines 24-67, col. 6, lines 1-67, col. 7, lines 1-67, col. 8, lines 1-67, col. 9, lines 2-67, col. 10, lines 1-67, col. 11, lines 1-62, col. 12, lines 5-19 and lines 48-67, col. 13, lines 1-61, and col. 14, lines 1-11.	1
Y, P	US 5,953,017 A (BEACH et al) 14 September 1999, col. 1, lines 36-67, col. 2, lines 1-8 and 38-54, col. 5, lines 50-67, col. 6, lines 1-48, col. 8, lines 10-67, col. 9, lines 1-23 and 59-67, col. 10, lines 1-67, col. 11, lines 1-41, col. 13, lines 19-34, col. 14, lines 53-67, col. 15, lines 1-20, col. 16, lines 60-67, col.17, lines 1-13, col. 18, lines 1-9 and lines 38-50, col. 19, lines 18-30, and col. 20, lines 6-20.	1
Y, E	US 6,055,515 A (CONSENTINO et al) 25 April 2000, col. 3, lines 25-67, col. 4, lines 7-10 and 40-54, col. 5, lines 2-13 and lines 27-39, col. 7, lines 23-58, col. 8, lines 31-67, and col. 9, lines 1-3.	1

BNSDOCID: <WO_____0057303A1_I_>